GPU Programming

To develop code for use on the GPU nodes, you can choose one of three programming models: the directive-based OpenACC model, the language-based CUDA model, or the language-based OpenCL model. Each model is described in this article.

For specific information about NVIDIA GPUs and the programming models for them, see the NVIDIA Developer Zone.

The Directive-Based OpenACC Model

OpenACC is a programming standard developed by PGI, CRAY, CAPS, and NVIDIA for offloading programs written in C, C++, and Fortran from a host CPU to an attached accelerator device.

Similar to OpenMP, OpenACC requires you to annotate your source code with directives describing sections of your code and corresponding data that are to be executed on the GPU:

- For C, the directive is **`#pragma acc ...`**
- For Fortran, the directive is **`!$acc ...`**

To compile your code, you need to use a PGI compiler and options. For example:

```
module use /nasa/modulefiles/testing
module load comp-pgi/20.4

pgcc -acc -fast -Minfo your_program.c
```

See PGI Accelerator Compilers with OpenACC Directives for more information.

The Language-Based CUDA Model

The Compute Unified Device Architecture (CUDA) was created by NVIDIA and implemented on the NVIDIA GPUs. With CUDA, code developers write programs in C, C++, and Fortran, and incorporate CUDA extensions and optimized libraries. You can use either of the following programming environments:

- CUDA C is the original CUDA programming environment developed by NVIDIA for GPUs. It allows direct programming of the GPU from a high-level language. If you choose to write or rewrite portions of your code in CUDA C, you will need to load a **`cuda`** module and use the NVIDIA CUDA C compiler (**`nvcc`**) to build the executable. For example,

  ```
  module load cuda/10.2
  nvcc your_program.cu
  ```
- CUDA Fortran, a product of a collaboration between PGI and NVIDIA, includes a Fortran 2003 compiler and tool chain for programming NVIDIA GPUs using Fortran. To use CUDA Fortran, you will need to load a PGI compiler module and use the compiler **`pgfortran`** to build the executable. For example:

  ```
  module use /nasa/modulefiles/testing
  module load comp-pgi/20.4

  pgfortran your_program.cuf
  ```

  If the filename of your program does not end with the **`.cuf`** or **`.CUF`** suffix, use the **`-Mcuda`** option. For example:

```
pgfortran -Mcuda your_program.xx
```

See CUDA Fortran for more information.

TIP: There are many example codes for various disciplines available in the **/nasa/cuda/10.2/samples** directory on Pleiades. If you want to learn more about CUDA programming, the following resources may also be helpful: the online CUDA programming guide, and the book CUDA by Example: An Introduction to General-Purpose GPU Programming.

The Language-Based OpenCL Model

The Open Computing Language (OpenCL) is a framework for writing programs that run across heterogeneous platforms including CPU, GPU, and others. It consists of the OpenCL C language specification and OpenCL runtime API specification. Although OpenCL is C-based, there has been further development to bridge OpenCL and Fortran, such as FortranCL and CLFORTRAN.

Here is an example of building an executable where the source code for the host, written in C++, reads a code for the GPU that is written in OpenCL:

```
g++ -c -I /nasa/cuda/10.2/include host_program.cpp -o your_program.o
g++ host_program.o -o host_program.exe -L /nasa/cuda/10.2/lib64 -lOpenCL
```

If you have any questions or need assistance, contact the NAS Control Room at (800) 331-8737, (650) 604-4444, or support@nas.nasa.gov.

Article ID: 647
Last updated: 16 Jul, 2020
Revision: 6
Systems Reference -> GPU Nodes -> GPU Programming
https://www.nas.nasa.gov/hecc/support/kb/entry/647/